APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/939,232	08/24/2001	William Joseph Armstrong	IBM / 182	4082

			EXAMINER
26517	7590	02/15/2006	PROCTOR, JASON SCOTT

WOOD, HERRON & EVANS, L.L.P. (IBM)
2700 CAREW TOWER
441 VINE STREET
CINCINNATI, OH 45202

ART UNIT	PAPER NUMBER
2123	

DATE MAILED: 02/15/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

	Application No.	**Applicant(s)**
Office Action Summary	09/939,232	ARMSTRONG ET AL.
	Examiner	**Art Unit**	
	Jason Proctor	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on <u>06 December 2005</u>.

2a)☒ This action is **FINAL.**   2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-19 and 21* is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19 and 21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on <u>24 January 2004</u> is/are: a)☒ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1)☒ Notice of References Cited (PTO-892)

2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3)☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date <u>*See attached.*</u>

4)☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5)☐ Notice of Informal Patent Application (PTO-152)

6)☐ Other: _____ .

Information Disclosure Statements
3/7/2005,
5/18/2005,
6/8/2005,
9/19/2005

DETAILED ACTION

Claims 1-19 and 21 are currently pending in the application. Claims 1, 11 and 19 have been amended and claim 20 has been cancelled by amendment submitted on 8 June 2005. Claims 1-19 and 21 have been rejected.

Information Disclosure Statements

The Examiner thanks Applicants for bring to attention the Information Disclosure Statements submitted on 7 March 2005, 18 May 2005, 8 June 2005, and 19 September 2005. These submissions have been considered by the Examiner.

Claim Objections

1.   Claim 19 is objected to because of the following informalities: The Examiner thanks Applicants for amending claim 19 in response to the suggestions made in the previous Office action to overcome the rejection under 35 U.S.C. § 101. That rejection has been withdrawn below. However, while the intended claimed invention is now believed to fall into a statutory category of invention, the language of the claim complicates this analysis.

Specifically, the claim recites "a program resident in a hypervisor," where a hypervisor is interpreted as being a particular computer program, i.e. "a program resident in a program". The claim concludes with "a tangible signal bearing medium bearing the first program." It is unclear if "the first program" refers to the program which is "resident in a hypervisor" or the combined "program resident in a hypervisor." If the former, there appears to be no computer readable

medium storing the specific "program resident in a hypervisor," while perhaps a copy of that program is borne by "a tangible signal bearing medium," which presents difficulties under 35 U.S.C. § 101. If the latter, clarification in the language of the claim is respectfully requested.

Whatever the analysis, the Examiner respectfully suggests considering more direct claim language. For example, a preamble stating, "A program product embodied on a computer readable medium executed by a computer, comprising:" accompanied by the deletion of "a tangible signal bearing medium bearing the first program" would improve the clarity and precision of the intended claim scope immensely.

Appropriate correction is required.

Claim Rejections - 35 USC § 101

The previous rejections under 35 U.S.C. § 101 have been withdrawn.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* v. *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1.   Determining the scope and contents of the prior art.

2.   Ascertaining the differences between the prior art and the claims at issue.

3.   Resolving the level of ordinary skill in the pertinent art.

4.   Considering objective evidence present in the application indicating obviousness or nonobviousness.

2.   Claims 1-19 and 21 are rejected under 35 U.S.C. § 103(a) as being unpatentable over US Patent No. 5,404,563 to Green et al. (Green) in view of US Patent No. 5,872,963 to Bitar et al. (Bitar), and further in view of "Operating System Concepts, Fifth Edition" by Abraham Silberschatz and Peter Baer Galvin (Silberschatz).

Green provides background knowledge of what is known in the art regarding a hypervisor on a logically partitioned computer system (column 1, lines 10-54). In particular, Green teaches, "The hypervisor 112 schedules, or allocates, the physical hardware components 104 to the logical partitions 114. For example, during a particular time-slice, the hypervisor 112 may allocate the physical CPU 106A to operate with the logical partition 114A. Specifically, the hypervisor 112 may dispatch the logical CPU 116B on the physical CPU 106A." (column 1, lines 33-43). Green does not disclose or teach the claimed scheduling scheme.

Bitar discloses the claimed scheduling scheme as a method for context switching between execution entities (abstract) wherein the execution entities may be virtual processors (column 1, lines 34-40; column 1, lines 55-63) but are equivalently described by Bitar as threads [*"A virtual processor may be a process, [...] a kernel thread, [...] or some other abstraction."* (column 1, lines 34-40); *"This abstraction, a virtual processor, is scheduled by the operating system scheduler for execution on available physical processors."* (column 1, lines 55-63)].

Bitar's method of switching between entities comprises a yielding virtual processor [*"threads which have finished their work"*, (column 10, line 48 – column 11, line 10)] which designates a target virtual processor [*"can transfer control of their respective processors to the preempted threads, thus resuming them."* (column 10, line 48 – column 11, line 10); *"thread B can transfer its processor to thread A, thus resuming thread A"* (column 11, lines 33-41)] and switching-in the target virtual processor for execution by the CPU in response to the requested yield [*"resuming thread them"*; *"resuming thread A"*; *supra*].

It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention to combine the teachings of Bitar regarding designating a target virtual processor by a yielding virtual processor on a logically partitioned computer system using a hypervisor because Green expressly teaches that the hypervisor schedules virtual processors and Bitar teaches an improved scheduling algorithm for dealing with situations such as spin locks (as in column 11, lines 33-41). The combination could be formed by implementing the processor control transfer method taught by Bitar in the logically partitioned hypervisor system of Green.

In response, Applicants argue primarily that:

> None of the cited prior art discloses the claimed feature of scheduling virtual processors (that, in turn, are used to schedule threads). For instance, the disclosure of Bitar et al. is limited to scheduling threads, not virtual processors. As outlined in Bitar et al., a virtual processor is not the same as a (user) thread (column 1, lines 32-33). [...] As claimed, a virtual processor is used to schedule threads (a scheduling entity), and it not an executing entity (as Bitar et al. defines threads at column 1, lines 17-19).

The Examiner respectfully traverses Applicants' argument as follows.

Applicants' allegation that Bitar teaches that "a virtual processor is not the same as a (user) thread" must be considered in context. From Bitar (column 1, lines 13-39):

A thread model of program execution has proven to be a viable method for parallel execution of program code both in single and multiprocessor machines. Under the thread model, programs are partitioned (by the user of by a compiler) into a set of parallel activities. The instantiation of each activity during execution of the program code is called a thread; if the program code includes more than one thread, the program is said to be multi-threaded. By partitioning the program code into threads, it is possible to create a more easily maintainable, more easily understood, and, possibly, faster program.

The thread abstraction described above is often referred to as a user-level thread. It is the entity that a user will create, using a threads interface, in order to express parallelism in a program. The operating system will provide a unit of scheduling, a virtual processor, to which a user-level thread will be mapped; the mapping may be performed statically, or dynamically when executing. This virtual processor will in turn be mapped to a physical processor by the operating system scheduler. Conceptually, it is useful to distinguish the user-level thread from the virtual processor.

A virtual processor may be a process, such as that provided by traditional UNIX systems, a kernel thread, such as that provided by Mach, or some other abstraction. It is the definition of the virtual processor and the related mapping of user-level threads to virtual processor that defines the performance characteristics of a threads implementation.

Bitar is clearly describing *the performance characteristics of a threads implementation*. In this context, Bitar makes a *conceptual* distinction between a *user-level thread* and a virtual processor. Why does Bitar make a *conceptual* distinction rather than a plain distinction? Perhaps so that the Bitar reference does not contradict itself by stating that **a virtual processor may be a process, kernel thread, or other abstraction** (column 1, lines 34-36).

A virtual processor may be a process, such as that provided by traditional UNIX systems, a kernel thread, such as that provided by Mach, or some other abstraction.

Applicants' parentheses ["a virtual processor is not the same as a (user) thread"] misrepresent the text of Bitar. Where, for example, would Applicants find support in the Bitar reference that a virtual processor not the same as a process or a kernel thread? Of course, Bitar states (column 1, lines 34-36).

A virtual processor may be a process, such as that provided by traditional UNIX systems, a kernel thread, such as that provided by Mach, or some other abstraction.

From where do Applicants' find support for the conclusion that "the disclosure of Bitar et al. is limited to scheduling threads, not virtual processors?" Bitar clearly contemplates the scheduling of virtual processors.

> This virtual processor will in turn be mapped to a physical processor by the operating system scheduler.
> (column 1, lines 30-32)
>
> Every operating system exports an abstraction that represents the basic unit of scheduling. [...] This
> abstraction, a virtual processor, is scheduled by the operating system scheduler for execution on available
> physical processors. (column 1, lines 55-63)

Additionally, the following portion of Applicants' arguments define an inoperable

invention as best understood by the Examiner:

> As claimed, a virtual processor is used to schedule threads (a scheduling entity), and it not an executing
> entity (as Bitar et al. defines threads at column 1, lines 17-19).

If the virtual processor [is] not an executing entity, that is, it does not execute, then how does the

virtual processor schedule threads? The Examiner respectfully submits that Applicants' appear

to have misconstrued the term "scheduling entity" to mean "an entity that schedules as opposed

to executing," rather than the more appropriate definition "an entity which is scheduled by a

scheduler." The act of scheduling clearly requires execution, and execution clearly requires

being mapped to a physical processor. If Applicants' claimed invention is distinguished from the

prior art on the basis of a virtual processor that does not execute, the claimed "virtual processor"

appears to be inoperative.

Applicants further argue that:

> Notably, Bitar et al. seeks to have threads executed by the CPU without using the virtual processor (or
> other scheduling entities, i.e., kernel thread) for efficiency considerations (column 5, lines 31-33 and lines
> 55-58).

It is unknown how this is notable. The claims do not present any limitations related to

efficiency, nor has the Examiner relied upon teachings related this particular efficiency in the

rejection. If Applicants' regard a certain efficiency to be critical to their disclosed invention, the

Examiner respectfully suggests positively recited claim limitations that reflect this feature.

Applicants further argue that:

> Perhaps some confusion has arisen due to the disclosure in Bitar et al. that describes a kernel thread in a Mach operating system as being the equivalent of a virtual processor in other operating systems (column 1, lines 54-column 2, line 5). However, Bitar et al. characterizes both the kernel thread and the virtual processor as being "the basic unit of scheduling." What these units are being used to schedule, or map, are user threads. As the claims now explicitly recite the function of the virtual processor as a entity used for scheduling, Applicants respectfully submit that the functional distinction between a virtual processor and a thread is patently clear.

The Examiner agrees that some confusion has arisen. In contrast, the teachings of Bitar are clear, and have been shown above to refute Applicants' analysis of the reference.

Bitar clearly teaches a versatile definition of *virtual processor*, **including** clear examples that a **virtual processor may be a process** and a **virtual processor may be a kernel thread**. Bitar clearly teaches **scheduling a virtual processor for execution on physical processors**. Considered as a whole, Bitar clearly renders obvious the notion of context switching between a variety of *execution entities*, (such as threads) **which may be virtual processors**. Of course, Bitar's teachings regarding context switching without using protected kernel mode have not been relied upon by the Examiner.

Applicants' arguments attempt to redefine the text of the Bitar reference to somehow exclude the claimed invention. There is no support for this interpretation in the Bitar reference.

Regardless of Applicants' analysis of the Bitar reference, the following argument is wholly unpersuasive:

> None of the cited prior art discloses the claimed feature of scheduling virtual processors (that, in turn, are used to schedule threads).

Applicants' arguments refer to an inherent function of virtual processors in the cited references. The prior art, as shown above, clearly discloses scheduling virtual processors. If those

processors are not used to schedule threads, the prior art would be inoperative. Indeed Bitar

expressly states:

> The operating system will provide a unit of scheduling, a virtual processor, to which a user-level thread will
> be mapped; the mapping may be performed statically, or dynamically when executing. This virtual
> processor will in turn be mapped to a physical processor by the operating system scheduler. (column 1,
> lines 27-32)

Clearly, Bitar discloses that a thread is mapped to a virtual processor, which then schedules that

thread to execute on a physical processor. If the virtual processor did not perform this function,

"mapping a thread to a virtual processor" would be synonymous with "destroying the thread,"

for it would never execute.

As an additional example of the inherency of this limitation, Applicants' attention is

respectfully Silberschatz, which has been cited merely to provide an example of the inherency of

using a virtual processor to schedule threads.

Silberschatz teaches on page 74:

> This layered approach is taken to its logical conclusion in the concept of a *virtual machine*. The VM
> operating system for IBM systems is the best example of the virtual-machine concept, because IBM
> pioneered the work in this area.
>   By using CPU scheduling (Chapter 5) and virtual-memory techniques (Chapter 9), an operating
> system can create the illusion of multiple processes, each executing on its own processor with its own
> (virtual) memory. ... Each process is provided with a (virtual) copy of the underlying computer (Figure
> 3.12).
>   The resources of the physical computer are shared to create the virtual machines. CPU scheduling
> can be used to share the CPU and to create the appearance that users have their own processor.

It is noted that the assignee for this application is "International Business Machines

Corporation," presumably the same "IBM" referred to by Silberschatz. If the virtual machine

and integral virtual processor did not schedule threads, it would be impossible for that virtual

machine to "create the appearance that users have their own processor," because scheduling

threads is inherent to the operation of a processor. Even a single-threaded system must schedule

the single thread to operate correctly.

Perhaps the Examiner misunderstands Applicants' argument. It would help to know

what, in Applicants' analysis, a prior art virtual processor does aside from schedule threads.

Applicants further argue that:

> While Green et al. introduces the general concept of hypervisors, the Examiner notes that Green does not
> disclose the claimed use of virtual processors to schedule threads.

It is unknown to what statements made by the Examiner Applicants are referring. It is

particularly unusual that "the claimed use of virtual processors to schedule threads" is a newly

presented amendment to the claim language.

Applicants' arguments have been fully considered but have been found unpersuasive.

The rejections of claims 2-19 and 21 incorporate the rejection and combination formed

above in regard to claim 1.

Regarding claim 2, Bitar et al. teaches a method of context switching wherein the target

virtual processor requires access to the CPU, wherein the yielding virtual processor controls the

CPU (column 10, line 48 – column 11, line 10; column 11, lines 33-41).

Regarding claim 3, Bitar et al. teaches a method of context switching comprising generating a yield command from the virtual processor, wherein the yield command includes pointer and status information regarding the target virtual processor (column 10, lines 9-33).

Regarding claim 4, Bitar et al. teaches a method of context switching comprising assigning status information to the target virtual processor (column 10, lines 9-33).

Regarding claim 5, Bitar et al. teaches a method of context switching comprising assigning a target count to the target virtual processor (column 10, lines 9-33). The preempt bit vector holds a value of 0 for a thread that has its resource requirements fulfilled and holds a value of 1 for a thread that has been preempted and requires resources to continue.

Regarding claim 6, Bitar et al. teaches a method of context switching comprising comparing the target count to a presented count conveyed in the yield request (column 10, lines 9-33; column 13, line 53 – column 14, line 24; column 16, lines 29-44).

Regarding claim 7, Bitar et al. teaches a method of context switching comprising aborting the yield in response to a yield-to-active command. If the processor is not needed, it will be reallocated to another process (column 16, lines 29-44).

Regarding claim 8, Bitar et al. teaches a method of context switching comprising designating the yielding virtual processor as waiting for the target virtual processor (column 10, line 48 – column 11, line 10; column 11, lines 33-41; column 16, lines 29-44).

Regarding claim 9, Bitar et al. teaches a method of context switching comprising designating the target virtual processor as having a yielding processor waiting for the target virtual processor (column 13, line 53 – column 14, line 24; column 16, lines 29-44).

Regarding claim 10, Bitar et al. teaches a method of context switching comprising storing the state of the yielding virtual processor (column 10, lines 9-33; column 13, lines 53-60).

Claims 11-18 are directed toward an apparatus comprising a computer system and a computer program to execute the method of claims 1-3, and 5-9. As the invention of Bitar et al. is a computer system and program (abstract), claims 11-18 are rejected for reasons similar to those given for claims 1-3, and 5-9 above.

Claim 19 is directed toward a program product and tangible signal bearing medium bearing a computer program which executes the method of claim 1. As the invention of Bitar et al. can be realized with a computer program, whether transmitted via a network or stored locally (abstract; column 17, lines 20-27; column 20, line 49 – column 21, line 10), claims 19 and 20 are rejected for reasons similar to those given for claim 1 above.

Regarding claim 21, Bitar teaches a user-level scheduler which includes a queue (schedule) used to schedule ready-to-run threads (column 8, lines 21-31). It is inherent that a user-level scheduler is a thread, therefore a virtual processor.

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Proctor whose telephone number is (571) 272-3713. The examiner can normally be reached on 8am-4pm M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Leo Picard can be reached at (571) 272-3749. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Jason Proctor
Examiner
Art Unit 2123

jsp

Paul P. Rodriguez   2/10/06
Primary Examiner
Art Unit 2125